Ready for Rain?

Herbst, Nikolas; Krebs, Rouven; Oikonomou, Giorgos; Kousiouris, George; Evangelinou, Athanasia; Iosup, Alexandru; Kounev, Samuel

published in
arXiv
2016

Technical Report: SPEC-RG-2016-01
Version: 1.0

Ready for Rain? A View from SPEC Research on the Future of Cloud Metrics

SPEC RG Cloud Working Group

Nikolas Herbst
Chair for Software Engineering
University of Würzburg
Würzburg, Germany
nikolas.herbst@uni-wuerzburg.de

Rouven Krebs
SAP AG
Walldorf, Germany
rouven.krebs@sap.com

Giorgos Oikonomou
Faculty of Engineering, Mathematics and Computer Science
Delft University of Technology
Delft, Netherlands
g.oikonomou@student.tudelft.nl

George Kousiouris
School of Electrical and Computer Engineering
National Technical University of Athens
Athens, Greece
gkousiou@mail.ntua.gr

Athanasia Evangelinou
School of Electrical and Computer Engineering
National Technical University of Athens
Athens, Greece
aevang@mail.ntua.gr

Alexandru Iosup
Faculty of Engineering, Mathematics and Computer Science
Delft University of Technology
Delft, Netherlands
A.Iosup@tudelft.nl

Samuel Kounev
Chair for Software Engineering
University of Würzburg
Würzburg
samuel.kounev@uni-wuerzburg.de

Acknowledgements

The authors would like to thank Kai Sachs, Klaus Lange, and Manoj Karunakaran Nambiar

Contents

Executive Summary

In the past decade, cloud computing has emerged from a pursuit for a service-driven information and communication technology (ICT), into a significant fraction of the ICT market. Responding to the growth of the market, many alternative cloud services and their underlying systems are currently vying for the attention of cloud users and providers. Thus, benchmarking them is needed, to enable cloud users to make an informed choice, and to enable system DevOps to tune, design, and evaluate their systems. This requires focusing on old and new system properties, possibly leading to the re-design of classic benchmarking metrics, such as expressing performance as throughput and latency (response time), and the design of new, cloud-specific metrics.

Addressing this requirement, in this work we focus on four system properties:

(i) *elasticity* of the cloud service, to accommodate large variations in the amount of service requested,

(ii) *performance isolation* between the tenants of shared cloud systems,

(iii) *availability* of cloud services and systems, and the

(iv) *operational risk* of running a production system in a cloud environment.

Focusing on key metrics, for each of these properties we review the state-of-the-art, then select or propose new metrics together with measurement approaches. We see the presented metrics as a foundation towards upcoming, industry-standard, cloud benchmarks.

Keywords[1]**:**

Cloud Computing; Metrics; Measurement; Benchmarking; Elasticity; Isolation; Performance; Service Level Objective; Availability; Operational Risk

CCS - General and reference - Metrics;
CCS - General and reference - Surveys and overviews;
CCS - Computer systems organization - Architectures -
Distributed architectures - Cloud computing;
CCS - Software and its engineering - Extra-functional properties.

Trademark

[1]The used keywords are defined as part of The 2012 ACM Computing Classification System acm (2012).

1 Introduction

Cloud computing is a paradigm under which ICT services are offered "as a service", that is, on-demand, and with payment expected to match what is actually used. Over the last decade, cloud computing has become increasingly important for the information and communication technology (ICT) industry. Cloud applications already represent over 10% of the entire ICT market in Europe European Commission (2014), and likely a similarly significant fraction of the ICT market in North America, Middle East, and Asia. By 2017, over three-quarters of the global business and personal data may reside in cloud data-centers, according to a recent IDC report IDC (2013). This promising growth trend makes clouds an interesting new target for benchmarking, with the goal of comparing, tuning, and improving the increasingly large set of cloud-based systems and applications, and the cloud fabric itself. However, traditional benchmarking approaches may not be able to address the new cloud computing settings. In classical benchmarking, common system performance metrics are measured on well-defined, often well-behaved systems-under-test (SUTs). In contrast, cloud systems can be built out of a rich, yet volatile combination of infrastructure, platforms, and entire software stacks, which in turn can be built out of cloud systems and offered as cloud services. For example, Netflix currently streams video as a cloud service to millions of people world-wide, occupying a significant fraction of the download capacity of Internet home-users, and, simultaneously, uses the infrastructure and the content-distribution platform provided as cloud services by Amazon AWS. Key to benchmarking the rich tapestry that characterizes many cloud services and their underlying systems is the re-definition of traditional benchmarking metrics for cloud settings, and the definition of new metrics that are unique to cloud computing. This is the focus of our work, and the main contribution of this report.

Academic studies, concerned public reports, and even company white papers indicate that a variety of new operational and user-driven phenomena take place in cloud settings. We consider in this work four such phenomena. First, cloud systems have been asked to deliver an illusion of infinite capacity and capability, raising interesting questions of how to provide results under wildly varying workloads, and forcing cloud systems to appear perfectly *elastic*. Second, cloud services and systems have been shown to exhibit high performance variability Iosup et al. (2011), against which modern cloud users have requested protection (*performance isolation*). Third, increasingly more demanding users expect today that the *availability* of cloud services is nearly perfect, and even few unavailability events can cause significant reputation and pecuniary damage to a cloud provider. Fourth, as the risks of not meeting implicit user-expectations and explicit service contracts (service level agreements, SLAs) are increasing with the scale of cloud operations, cloud providers have become increasingly more interested to reduce their *operational risk*.

With the market growing and maturing, many cloud services are now competing for the attention of existing and new cloud users. Thus, quantifying the capabilities of the system features that respond to various cloud phenomena, and in particular benchmarking the non-functional properties of cloud systems (including performance), is increasingly important. We ask in this work three important research questions: *Do traditional metrics already support the cloud features created to match the four phenomena we consider in this work?* As we report in this work, the survey of the state-of-the-art indicates that the answer to this question is "No.", we further raise the follow-up research question: *Which new metrics are needed, to support the cloud features created to match the four phenomena we consider in this work?* To this question we cannot offer a definitive answer, but our results include various new metrics and adaptations of existing metrics that may lead to new industrial-grade benchmarks.

Addressing the two research questions, the goal of this report is to lay a foundation for making various cloud offerings and technologies comparable to each other, and provide a common

understanding among all cloud stakeholders. Although elasticity, isolation, availability, and operational risk are already perceived as important aspects in the academia and by the industry, they have never before been thoroughly defined and surveyed. As we show in this work, their meaning can be different for different stakeholders, and in some cases existing definitions are inconsistent or even contrary to each other. Toward reaching our goal, our main contribution is four-fold. Each contribution is focusing on the foundations of benchmarking one feature of cloud systems. In turn, the features are:

1. *Elasticity*, addressed in Section 2. Elasticity offers the opportunity to automatically adapt the resource supply to a changing demand. The quality of elastic adaptation is only indirectly captured by traditional performance metrics, such as response time and utilization, and requires new approaches. We present in this work a set of metrics and methods for combining them to capture the accuracy and timing aspects of elastic platforms.

2. *Performance isolation*, addressed in Section 3. The underlying cloud infrastructure has the important task to isolate different customers sharing the same hardware from each other with regards to the performance they observe. We present in this work metrics that capture the influence of disruptive workloads, the maximum disruptive load that a system can handle, and the degree of breaches of the performance isolation agreement.

3. *Availability*, addressed in Section 4. To quantify the availability of their business critical cloud applications and compare them for different contexts, in this work we analyze the availability definitions used by various cloud providers. We then define a simple metric of SLA adherence that enables direct comparisons between providers with otherwise different definitions of availability.

4. *Operational risk-related*, addressed in Section 5. And on a more general level than the other features, we also focus in this work on estimating different types of operational risks that are connected with running software in the cloud. We define here various relevant metrics, and a measurement methodology that addresses them.

2 Elasticity

2.1 Goal and Relevance

Elasticity has originally been defined in physics as a material property capturing the capability of returning to its original state after a deformation. In economics, elasticity captures the effect of change in one variable to another dependent variable. In both cases, elasticity is an intuitive concept and can be precisely described using mathematical formulas.

The concept of elasticity has been transferred to the context of cloud computing and is commonly considered as one of the central attributes of the cloud paradigm as in Plummer et al. (2009). For marketing purposes, the term elasticity is heavily used in cloud providers' advertisements and even in the naming of specific products or services. Even though tremendous efforts are invested to enable cloud systems to behave in an elastic manner, no common and precise understanding of this term in the context of cloud computing has been established so far, and no ways have been proposed to quantify and compare elastic behavior.

2.2 Prerequisites

The scalability of a system including all hardware, virtualization, and software layers within its boundaries is a prerequisite in order to be able to speak of elasticity. Scalability is the ability of a system to sustain increasing workloads with adequate performance provided that hardware resources are added. Scalability in the context of distributed systems has been defined in Jogalekar and Woodside (2000), as well as more recently in Duboc (2009); Duboc et al. (2007), where also a measurement methodology is proposed.

Given that elasticity is related to the ability of a system to adapt to changes in workloads and resource demands, the existence of at least one specific adaptation process is assumed. The latter is normally automated, however, in a broader sense, it could also contain manual steps. Without a defined adaptation process, a scalable system cannot behave in an elastic manner, as scalability on its own does not include temporal aspects.

When evaluating elasticity, the following points need to be checked beforehand:

- *Autonomic Scaling:*
  What adaptation process is used for autonomic scaling?
- *Elasticity Dimensions:*
  What is the set of resource types scaled as part of the adaptation process?
- *Resource Scaling Units:*
  For each resource type, in what unit is the amount of allocated resources varied?
- *Scalability Bounds:*
  For each resource type, what is the upper bound on the amount of resources that can be allocated?

2.3 Definition

Elasticity is the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources *match* the current demand as closely as possible.

Dimensions and Core Aspects Any given adaptation process is defined in the context of at least one or possibly multiple types of resources that can be scaled up or down as part of the adaptation. Each resource type can be seen as a separate dimension of the adaptation process

with its own elasticity properties. If a resource type is a container of other resources types, like in the case of a virtual machine having assigned CPU cores and RAM, elasticity can be considered at multiple levels. Normally, resources of a given resource type can only be provisioned in discrete units like CPU cores, virtual machines (VMs), or physical nodes. For each dimension of the adaptation process with respect to a specific resource type, elasticity captures the following core aspects of the adaptation:

Timing The timing aspect is captured by the percentages a system is in an under-provisioned, over-provisioned or perfect state and by the amount of superfluous adaptations.

Accuracy The accuracy of scaling is defined as the average absolute deviation of the current amount of allocated resources from the actual resource demand.

As discussed above, elasticity is always considered with respect to one or more resource types. Thus, a direct comparison between two systems in terms of elasticity is only possible if the same resource types (measured in identical units) are scaled.

To evaluate the actual observable elasticity in a given scenario, as a first step, one must define the criterion based on which the amount of provisioned resources is considered to *match* the actual current demand needed to satisfy the system's given performance requirements. Based on such a matching criterion, specific metrics that quantify the above mentioned core aspects, as discussed in more detail in Section 2.6, can be defined to quantify the practically achieved elasticity in comparison to the hypothetical *optimal elasticity*. The latter corresponds to the hypothetical case where the system is scalable with respect to all considered elasticity dimensions without any upper bounds on the amount of resources that can be provisioned and where resources are provisioned and de-provisioned immediately as they are needed exactly matching the actual demand at any point in time. *Optimal elasticity*, as defined here, would only be limited by the resource scaling units.

Differentiation This paragraph discusses the conceptual differences between elasticity and the related terms scalability and efficiency.

Scalability is a prerequisite for elasticity, but it does not consider temporal aspects of how fast, how often, and at what granularity scaling actions can be performed. Scalability is the ability of the system to sustain increasing workloads by making use of additional resources, and therefore, in contrast to elasticity, it is not directly related to how well the actual resource demands are matched by the provisioned resources at any point in time.

Efficiency expresses the amount of resources consumed for processing a given amount of work. In contrast to elasticity, efficiency is not limited to resource types that are scaled as part of the system's adaptation mechanisms. Normally, better elasticity results in higher efficiency. The other way round, this implication is not given, as efficiency can be influenced by other factors independent of the system's elasticity mechanisms (e.g., different implementations of the same operation).

2.4 Derivation of the Matching Function

To capture the criterion based on which the amount of provisioned resources is considered to match the actual current demand, we define a matching function $m(w) = r$ as a system specific function that returns the minimal amount of resources r for a given resource type needed to satisfy the system's performance requirements at a specified workload intensity. The workload intensity w can be specified either as the number of workload units (e.g., user requests) present at the system at the same time (concurrency level), or as the number of workload units that

arrive per unit of time (arrival rate). A matching function is needed for both directions of scaling (up/down), as it cannot be assumed that the optimal resource allocation level when transitioning from an under-provisioned state (upwards) are the same as when transitioning from an over-provisioned state (downwards).

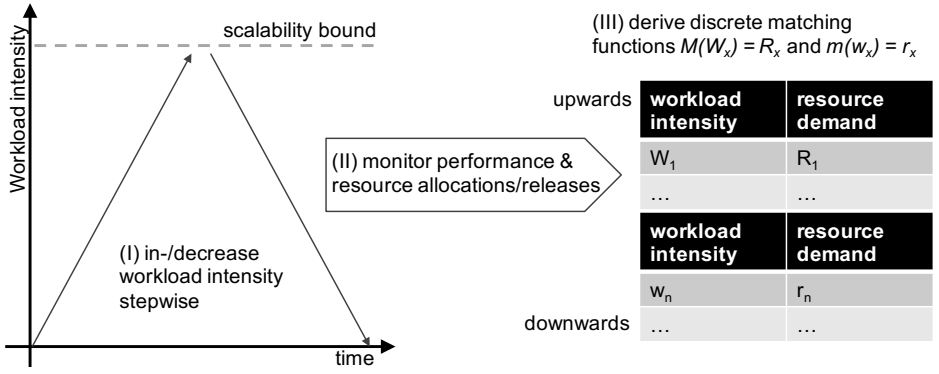


Figure 2.1: Illustration of a measurement-based derivation of matching functions

The matching functions can be derived based on measurements, as illustrated in Figure 2.1, by increasing the workload intensity w stepwise, while measuring the resource consumption r, and tracking resource allocation action changes. The process is then repeated for decreasing w. After each change in the workload intensity, the system should be given enough time to adapt its resource allocations reaching a stable state for the respective workload intensity. As a result of this step, a system specific table is derived that maps workload intensity levels to resource demands, and the other way round, for both scaling directions within the scaling bounds.

2.5 Related Elasticity Metrics

In this section, we group existing metrics and benchmark approaches for elasticity according to their perspective and discuss shortcomings.

Elasticity Metrics: Several metrics for elasticity have been proposed so far:
(i) The "scaling latency" metrics in Li et al. (2010, 2012b) or the "provisioning interval" in Chandler et al. (2012) capture the time to bring up or drop a resource. This duration is a technical property of elastic environments independent of demand changes and the elasticity mechanism itself that decides when to trigger a reconfiguration. Thus, these metrics are insufficient to fully characterize the elasticity of a platform.
(ii) The "elastic speedup" metric proposed by the SPEC OSG in Chandler et al. (2012) relates the processing capability of a system at different scaling levels. This metric does not capture any dynamic aspect of elasticity and is regarded as scalability metric.
(iii) The "reaction time" metric in Kuperberg et al. (2011) can only be computed if a unique mapping between resource demand changes and supply changes exists. This assumption does not hold especially for proactive or unstable elasticity mechanisms.
(iv) The approaches in Binnig et al. (2009); Cooper et al. (2010); Almeida et al. (2013); Dory et al. (2011) describe elasticity indirectly by analysing response times for significant changes or SLO compliance. In theory, perfect elasticity results in constant response times for varying arrival rates. In practice, detailed reasoning about the quality of platform adaptations based on response times is hampered due to black-box abstractions, e.g., that the amount of surplus

resources remains hidden and individual requests have inherently different response times.
(v) Cost-based metrics are proposed in Islam et al. (2012); Folkerts et al. (2012); Suleiman (2012); Weinman (2011); Tinnefeld et al. (2014) to quantify the impact of elasticity either by comparing the resulting costs to the costs for a peak-load static assignment of resources or the costs of a hypothetical perfect elastic platform. In both cases, the resulting metrics strongly depend on the underlying cost model as well as on the assumed penalty for under-provisioning and thus do not allow fair cross-platform comparison.
(vi) The integral-based "agility" metric proposed by the SPEC OSG in Chandler et al. (2012) compares the demand and supply over time normalized by the average demand. They state the metric becomes invalid in cases where SLOs are not met. This metric resembles the previously proposed "precision" metric in Herbst et al. (2013) and is included in a refined version normalized by time in this document (see Section 2.6) to capture the accuracy aspect of elastic adaptations also in situations when SLOs are not met.

Benchmarks: Existing benchmark approaches for elasticity as in Folkerts et al. (2012); Suleiman (2012); Weinman (2011); Shawky and Ali (2012); Islam et al. (2012); Dory et al. (2011); Almeida et al. (2013); Tinnefeld et al. (2014); Cooper et al. (2010) account neither for differences in efficiency of the underlying physical resources nor for possibly non-linear scalability of the platform. As a consequence, the quantification of resource elasticity is not realized in isolation of these related platform attributes as previously highlighted in Herbst et al. (2013). In contrast, our proposed approach uses the insights gained in an automated scalability and performance analysis to adapt the load profile in a platform specific way. In addition, existing benchmarks employ load profiles, which rarely cover the realistic variability of the load intensity over time. In several cases, the aspect of scaling downwards is omitted as in Dory et al. (2011); Shawky and Ali (2012). In Islam et al. (2012), sinus like load profiles with plateaus are employed. Real-world load profiles exhibit a mixture of seasonal patterns, trends, bursts and noise. We account for the generic benchmark requirement "representativeness" as mentioned in Huppler (2009) by employing the load profile modeling formalism DLIM presented in von Kistowski et al. (2014); von Kistowski (2014).

2.6 Proposed Elasticity Metrics

The *demand* of a certain load intensity is understood as the minimal amount of resources required for fulfilling a given performance related service level objective (SLO). The metrics are designed to characterize two core aspects of elasticity: *Accuracy* and *timing*[2]. For all metrics, the optimal value is zero and defines the perfect elastic system. For a valid comparison of elasticity based on the proposed set of metrics, the platforms (i) require the existence of an autonomic adaption process, (ii) the scaling of the same resource type, e.g., CPU cores or virtual machines (VMs) and (iii) within the same ranges, e.g., 1 to 20 resource units.

The metrics evaluate the resulting elastic behavior and thus are not designed for distinct descriptions of the underlying hardware, the virtualization technology, the used cloud management software or the used elasticity strategy and its configuration. As a consequence, the metric and the measurement methodology are applicable in situations where not all influencing factors are known. All metrics require two discrete curves as input: The demand curve, which defines how the resource demand varies during the measurement period, and the supply curve, which defines how the amount of actually used resource varies.

The following Section 2.6 describes the metrics for the *accuracy* aspect whereas Section 2.6 presents a set of metrics for the quantification of the *timing* aspect. In Section 2.7, we outline an

[2]In Herbst et al. (2013), these aspects are referred to as *precision* and *speed*.

approach for the aggregation of the proposed metrics enabling to compute a consistent ranking between multiple elastic cloud environments and configurations.

Accuracy

As visualized in Fig. 2.2, the under-provisioning accuracy metric $accuracy_U$ formally defined in the previous publication Herbst et al. (2013), is calculated as the sum of areas $\sum U$ where the resource demand exceeds the supply normalized by the duration of the measurement period T. Accordingly, the over-provisioning accuracy metric $accuracy_O$ bases on the sum of areas $(\sum O)$ where the resource supply exceeds the demand.

$$\text{Under-provisioning: } accuracy_U\ [resource\ units] = \frac{\sum U}{T} \tag{1}$$

$$\text{Over-provisioning: } accuracy_O\ [resource\ units] = \frac{\sum O}{T} \tag{2}$$

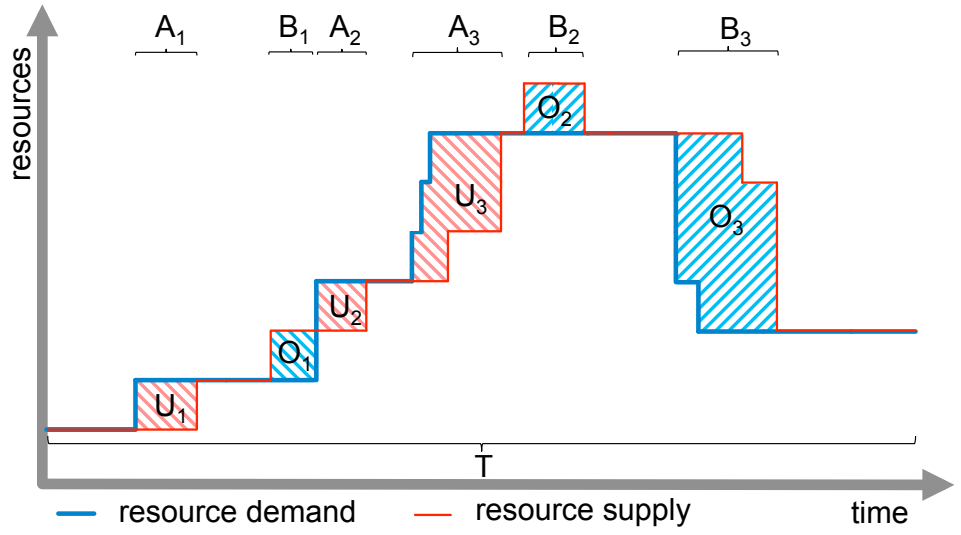


Figure 2.2: Illustration for the definition of accuracy and provisioning timeshare metrics.

Thus, $accuracy_U$ and $accuracy_O$ are the average of the absolute deviations between the current amounts of allocated resources and their respective actual resource demands. Since under-provisioning results in violating SLOs, a customer might want to use platform that do not tend to under-provision at all. Thus, the challenge for providers is to ensure that enough resources are provided at any point in time, but at the same time beat the competitors by over-provisioning not too much. Considering this, separate accuracy measures for over-provisioning and under-provisioning help providers to communicate their elasticity capabilities and customers to select a cloud provider according to their needs.

Timing

We highlight the *timing* aspect of elasticity from the viewpoints of the pure *provisioning timeshare* and the *jitter* accounting for superfluous or skipped adaptations.

Provisioning Timeshare The two accuracy metrics allow no reasoning whether the average amount of under-/over-provisioned resources results from a few big deviations between demand

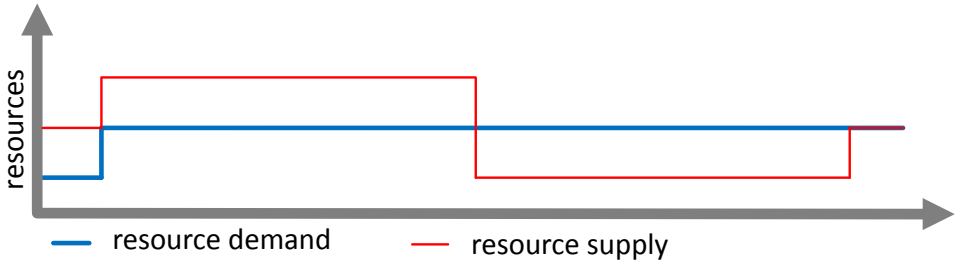
and supply or if it is rather caused by a constant small deviation. To address this, the following two metrics are designed to give more insights about the ratio of time in which under- or over-provisioning occurs.

As visualized in Fig. 2.2, the following metrics $timeshare_U$ and $timeshare_O$ are computed by summing up the total amount of time spend in an under- ($\sum A$) or over-provisioned ($\sum B$) state normalized by the duration of the measurement period. Thus, they measure the overall timeshare spent in under- or over-provisioned states:

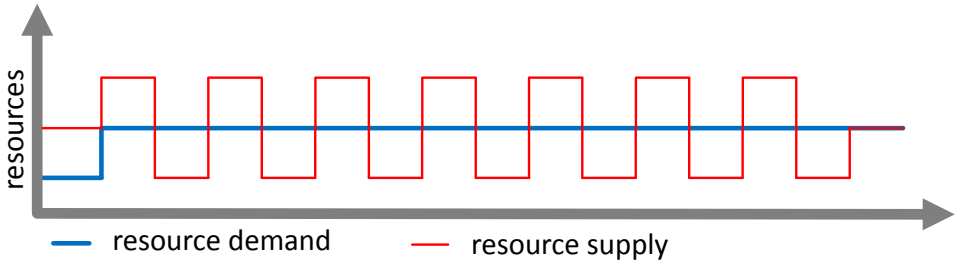
$$\text{Under-provisioning: } timeshare_U\ [\%] = \frac{\sum A}{T} \tag{3}$$

$$\text{Over-provisioning: } timeshare_O\ [\%] = \frac{\sum B}{T} \tag{4}$$

Jitter Although the *accuracy* and *timeshare* metrics measure important aspects of elasticity, platforms can still behave very different while producing the same metric values for *accuracy* and *timeshare* metrics. An example is shown in Figure 2.3.



(a) Platform A



(b) Platform B

Figure 2.3: Platforms with different elastic behaviors that produce equal results for *accuracy* and *timeshare* metrics

Both Platforms A and B exhibit the same accuracy metrics and spend the same amount of time in the under-provisioned respectively over-provisioned states. However, the behavior of both platforms differs significantly. Platform B triggers unnecessary resource supply adaptations whereas Platform A triggers just a few. We propose to capture this with a further metric called *jitter* to support reasoning for instance-hour-based pricing models as well as the operators view on minimizing adaptation overheads.

The *jitter* metric compares the amount of adaptations within the supply curve E_S with the number of adaptations within the demand curve E_D. If a platform de-/allocates more than one resource at a time, the adaptations are counted individually per resource unit. The difference is normalized with the length of the measurement period T:

$$\text{Jitter metric: } jitter\ \left[\frac{\#adap.}{min}\right] = \frac{E_S - E_D}{T}$$

A negative *jitter* metric indicates that the platform adapts rather sluggish to a changed demand. A positive *jitter* metric means that the platform tends to oscillate like Platforms A (little) and B (heavily) as in Figure 2.3. High absolute values of *jitter* metrics in general indicate that the platform is not able to react on demand changes appropriately. In contrast to the accuracy and timeshare metrics, a *jitter* value of zero is a necessary, but not sufficient requirement for a perfect elastic system.

2.7 Metric Aggregation

The sections above explained different metrics for capturing core aspects of elasticity:

- $accuracy_U$ and $accuracy_O$ measure average resource amount deviations
- $timeshare_U$ and $timeshare_O$ measure ratios of time in under- or over-provisioned states
- *jitter* measures the difference in demand and supply changes

For a more comfortable comparison of platforms, we propose a way to aggregate the metrics and to build a consistent and fair ranking, similar to the aggregation and ranking of results in established benchmarks, e.g., SPEC CPU2006[3].

Our proposed approach to compute an aggregated *elastic speedup* consists of the following three steps:

1. Aggregate the accuracy and timeshare sub metrics into a weighted accuracy and a weighted timeshare metric, respectively.
2. Compute elasticity speedups for both of the aggregated metrics using the metric values of a baseline platform.
3. Use the geometric mean to aggregate the speedups for *accuracy* and *timeshare* to a *elastic speedup* measure.

The resulting *elastic speedup* measure can be used to compare platforms without having to compare each elasticity metric separately. As a limitation of this approach, the *jitter* metric should not be included (as it can become zero also in realistic cases of imperfect elasticity). Each of the three steps is explained in the following.

1. The $accuracy_U$ and $accuracy_O$ metrics are combined to a weighted accuracy metric $accuracy_{weighted}$:

$$accuracy_{weighted} = w_{acc_U} \cdot accuracy_U + w_{acc_O} \cdot accuracy_O \tag{5}$$

$$\text{with } w_{acc_U}, w_{acc_O} \in [0,1], \quad w_{acc_U} + w_{acc_O} = 1 \tag{6}$$

   In the same way, the $timeshare_U$ and $timeshare_O$ metrics are combined to a weighted timeshare metric $timeshare_{weighted}$:

$$timeshare_{weighted} = w_{ts_U} \cdot timeshare_U + w_{ts_O} \cdot timeshare_O \tag{7}$$

$$\text{with } w_{ts_U}, w_{ts_O} \in [0,1], \quad w_{ts_U} + w_{ts_O} = 1 \tag{8}$$

2. Let x be a vector that stores the metric results:

$$x = (x_1, x_2) = (accuracy_{weighted}, timeshare_{weighted}) \tag{9}$$

[3]SPEC CPU2006: `http://www.spec.org/cpu2006/`

For a metric vector x_{base} of a given baseline platform and a metric vector x_k of a benchmarked platform k, the speedup vector s_k is computed as follows:

$$s_k = \left(s_{k_{accuracy}}, s_{k_{timeshare}}\right) = \left(\frac{x_{base_1}}{x_{k_1}}, \frac{x_{base_2}}{x_{k_2}}\right) \tag{10}$$

3. The elements of s_k are aggregated to an unweighted *elastic speedup*$_{unweighted_k}$ measure using the geometric mean:

$$\textit{elastic speedup}_{unweighted_k} = \sqrt{s_{k_{accuracy}} \cdot s_{k_{timeshare}}} \tag{11}$$

The geometric mean produces consistent rankings, no matter how measurements are normalized and is the only correct mean for normalized measurements Fleming and Wallace (1986). Thus, the ranking of the platforms according to *elastic speedup*$_k$ is consistent, regardless of the chosen baseline platform. Furthermore, different preferences concerning the elasticity aspects can be taken into account by using the weighted geometric mean for computing the *elastic speedup*$_{weighted_k}$:

$$\textit{elastic speedup}_{weighted_k} = {s_{k_{accuracy}}}^{w_{acc}} \cdot {s_{k_{timeshare}}}^{w_{ts}} \tag{12}$$

$$\text{with } w_{acc},\, w_{ts} \in [0,1], \quad w_{acc} + w_{ts} = 1 \tag{13}$$

The following equation summarizes the three steps:

$$\textit{elastic speedup}_{weighted_k} = \left(\frac{w_{acc_U} \cdot acc_{U\,base} + w_{acc_O} \cdot acc_{O\,base}}{w_{acc_U} \cdot acc_{U\,k} + w_{acc_O} \cdot acc_{O\,k}}\right)^{w_{acc}} \cdot \left(\frac{w_{ts_U} \cdot ts_{U\,base} + w_{ts_O} \cdot ts_{O\,base}}{w_{ts_U} \cdot ts_{U\,k} + w_{ts_O} \cdot ts_{O\,k}}\right)^{w_{ts}}$$
$$\text{with } w_{acc_U},\, w_{acc_O},\, w_{ts_U},\, w_{ts_O},\, w_{acc},\, w_{ts} \in [0,1];$$
$$w_{acc_U} + w_{acc_O} = 1; \quad w_{ts_U} + w_{ts_O} = 1; \quad w_{acc} + w_{ts} = 1$$

Elasticity Metric Weights A single number measuring the elasticity as shown in the equation above can be adjusted according to the preferences of the target audience by using different weights. For example, the accuracy weights $w_{acc_U} = 0.2, \;\; w_{acc_O} = 0.8$ allow to amplify the influence of the amount of over-provisioned resources compared to the amount of under-provisioned resources. A reason for doing so could be that over-provisioning leads to additional costs, which mainly depend on the amount of over-provisioned resources. The cost for under-provisioning in contrast does not depend that much on the amount of under-provisioned resources but more on how long the platform under-provisions. This observation can be taken into account by using timeshare weights like: $w_{ts_U} = 0.8, \;\; w_{ts_O} = 0.2$. Finally, when combining the *accuracy* and *timeshare* speed ups, the weights $w_{acc}, \; w_{ts}$ can help to prioritize different elasticity aspects. Here, weights like $w_{acc} = \frac{1}{3}, \;\; w_{ts} = \frac{2}{3}$ for example would double the importance short under- and over-provisioning periods compared to the importance of small under- or over-provisioning amounts.

2.8 Elasticity Measurement Approach

This paragraph shortly sketches an elasticity benchmarking concept that we propose as described in Herbst et al. (2015) together with its implementation called BUNGEE[4]. Generic and cloud specific benchmark requirements as stated in Huppler (2009, 2012) and Folkerts et al. (2012) are considered in this approach. Figure 2.4 shows the four main steps in the measurement process shortly explained in the following:

[4]BUNGEE Cloud Elasticity Benchmark: `http://descartes.tools/bungee`

1. **Platform Analysis:** The benchmark analyzes the system under test (SUT) with respect to the performance of its underlying resources and its scaling behavior.

2. **Benchmark Calibration:** The results of the analysis are used to adjust the load intensity profile injected on the SUT in a way that it induces the same resource demand on all compared platforms.

3. **Measurement:** The load generator exposes the SUT to a varying workload according to the adjusted load profile. The benchmark extracts the actual induced resource demand and monitors resource supply changes on the SUT.

4. **Elasticity Evaluation:** Elasticity metrics are computed and used to compare the resource demand and resource supply curves with respect to different elasticity aspects.

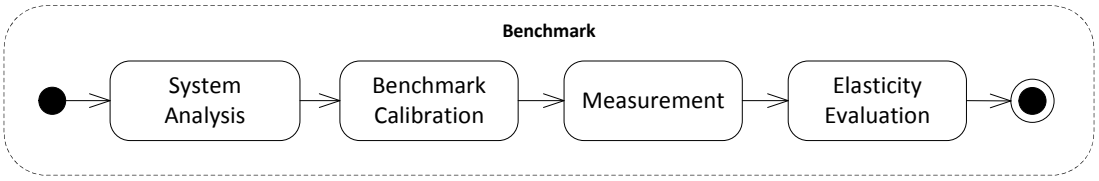


Figure 2.4: Activity diagram for the benchmark work flow

The results of an exemplary benchmark run are plotted in Figure 2.5 and the computed elasticity metrics in Table 2.1

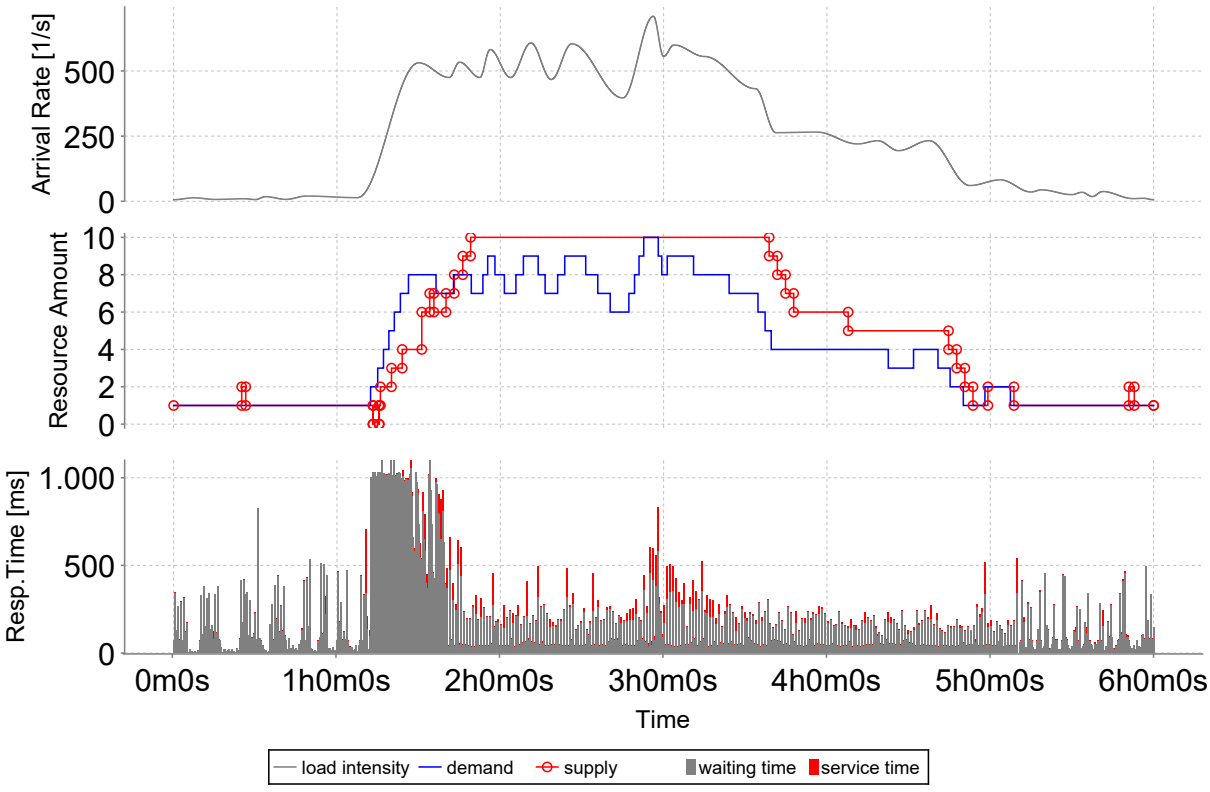


Figure 2.5: Plot resource demand and supply for an exemplary benchmark run on a public cloud

Table 2.1: Metric results for an exemplary benchmark run

acc_O [#res.]	acc_U [#res.]	ts_O [%]	ts_U [%]	$jitter$ $\left[\frac{\#adap.}{min}\right]$
1.053	0.180	51.9	8.1	-0.033

2.9 Discussion

This section presents a set of metrics for capturing the accuracy and timing aspects of elastic platforms. Existing cost-based and end-user focused metrics are strongly dependent on the provider's cost model and therefore not seen as independent for valid cross-platform comparisons. Low-level technical elasticity metrics like mean provisioning time leave out the impact of an auto-scaler configuration. We provide a metric aggregation method based speed-up ratios for relative comparisons and showed how it can be adapted to personal preferences using weights. We shortly describe a corresponding elasticity benchmarking methodology that enables cross-platform comparisons even if the performance of underlying resource units differs.

3 Performance Isolation

In this section, metrics and techniques for quantifying performance isolation based on current research being presented. Two different methodologies and several alternative metrics along with appropriate measurement techniques for quantifying the isolation capabilities of IT systems with help of performance benchmarks.

3.1 Goal and Relevance

Cloud Computing shares resources among several customers on various layers like IaaS, PaaS or SaaS. The isolating of cloud customers with regards to performance is one of the major challenges to achieve reliable performance.

The allocation of hardware resources is handled by the lower levels (e.g., infrastructure level) in the stack. Therefore, performance isolation is a bigger challenge in the upper levels (e.g., platform and software) as they intentionally have no direct resource control. Nevertheless, also on the infrastructure level we can observe significant influence between virtual machines as described in Huber et al. (2011).

Performance isolation is an important aspect for various stakeholders. When a developer or architect has to develop a mechanism to ensure performance isolation between customers they need to validate the effectiveness of their approach to ensure the quality of the product. Furthermore, to improve an existing mechanism they need an isolation metric to compare different variants of the solution. When a system owner has to decide for one particular deployment in a virtual environment not only traditional questions like the separation of components on various hosts are of importance but also how the deployments non-functional runtime properties will be influenced. the hypervisor with regards to resource allocation mechanism have to be considered. concerns might be important.

3.2 Foundations for the Metrics

The subsequently presented Metrics follow the ideas presented in Krebs et al. (2012, 2014).

Definition of Performance Isolation Performance concerns in cloud environments are a serious obstacle for consumers. To avoid distrust, it is necessary to ensure a fair behavior. This means, Customers working within their assigned quota should not suffer from customers exceeding their quotas. Quota refers to the amount of workload a customer is allowed to execute one metric might be the request rate.

Performance Isolation A system is performance-isolated, if for customers working within their quotas the performance is not affected when other customers exceed their quotas. A decreasing performance for the customers exceeding their quotas is accepted. It is possible to relate the definition to SLAs: A decreased performance for the customers working within their quotas is acceptable as long as it is within their SLAs.

Basic Idea The metrics defined may be applied to quantify the isolation of any measurable QoS-related system property in any system shared between different entities. Of course, the actual type of workload and QoS must be selected according to the scenario under investigation.

The metrics distinguish between groups of *disruptive* and *abiding* customers. The latter work within their given quota (e.g., defined number of requests/s) the former exceed their quota. Isolation metrics are based on the influence of the disruptive customers on the abiding customers. Thus we have two groups and observe the performance of one group as a function of the workload of the other group (cf. Figure 3.1).

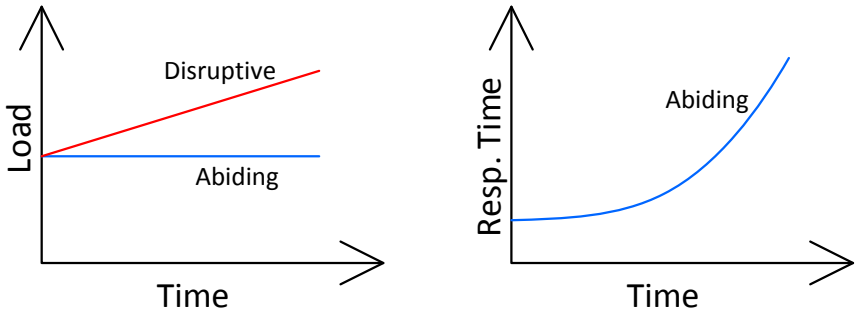


Figure 3.1: Influence of the disruptive tenant onto the abiding.

To provide a level playing field for comparisons, a description of a workload profile W is required together with the metrics. Although, the concrete definition of the workload W is a case to case decision and usually defined to answer a dedicated question, we share some general thoughts concerning this issue. Multi-tenant cloud applications (MTAs) are operated at rather high utilization for economic reasons. Another important reason to run the test system under high utilization, is the goal to evaluate performance isolation aspects. In a system with low utilization of the resources, the increased workload of one tenant has a low impact upon the performance of the others, as no bottleneck exists. Another aspect is related to existing guarantees. If the provider wants to maintain a certain QoS, it is possible to configure the overall reference workload in a way the average systems QoS is close to this value. In this case, a small increase of workload at the disruptive tenant, immediately results in violations for the abiding ones, in case of a weak isolation. In case no SLA based guarantee exists, and the bottleneck resource is unknown, a measurement to identify the system's maximum throughput, by increasing all tenants workloads in parallel is feasible. To increase the speed finding this point, a binary like search can be used. Usually this workload is accompanied with the highest possible utilization of the bottleneck resource. This means, that increasing the workload in a non-isolated system, will immediately result in less performance for all other tenants. Consequently, it would immediately cause guarantee violations. Another argument is, that an isolation mechanisms should intervene latest at this point. The key findings for the workload W are summarized as follows: (I) High load/utilization preferable, (II) QoS observed should be close to the guarantee, and (III) the systems maximum throughput can be used as an indicator.

3.3 Metrics based on QoS impact

For the definition of the metrics, a set of symbols are defined in Table 3.1.

These metrics depend on at least two measurements. First, the observed QoS results for every $t \in A$ at a reference workload W_{ref}. Second, the results for every $t \in A$ at a workload W_{disr} when a subset of the customers have increased their load to challenge the system's isolation mechanisms. As previously defined W_{ref} and W_{disr} are composed of the workload of the same set of customers which is the union of A and D. At W_{disr} the workload of the disruptive customers is increased.

We consider the relative difference of the QoS (Δz_A) for abiding customers at the reference workload compared to the disruptive workload.

$$\Delta z_A = \frac{\sum_{t \in A} [z_t(W_{disr}) - z_t(W_{ref})]}{\sum_{t \in A} z_t(W_{ref})} \tag{14}$$

Symbol	Meaning
t	A customer in the system.
D	Set of disruptive customers exceeding their quotas (e.g., contains customers inducing more than the allowed requests per second). $\|D\| > 0$
A	Set of abiding customers not exceeding their quotas (e.g., contains customers inducing less than the allowed requests per second).$\|A\| > 0$
w_t	Workload caused by customer t represented as numeric value $\in \mathbb{R}_0^+$. The workload is considered to increase with higher values (e.g., request rate and job size). $w_t \in W$
W	The total system workload as a set of the workloads induced by all individual customers. Thus, the load of the disruptive and abiding ones.
$z_t(W)$	A numeric value describing the QoS provided to customer t. The individual QoS a customer observes depends on the composed workload of all customer W. We consider QoS metrics where lower values of $z_t(W)$ correspond to better qualities (e.g., response time) and $z_t(W) \in \mathbb{R}_0^+$
I	The degree of isolation provided by the system. An index is added to distinguish different types of isolation metrics. The various indices are introduced later.

Table 3.1: Overview of variables and symbols

Additionally, we consider the relative difference of the load induced by the two workloads.

$$\Delta w = \frac{\sum\limits_{w_t \in W_{disr}} w_t - \sum\limits_{w_t \in W_{ref}} w_t}{\sum\limits_{w_t \in W_{ref}} w_t} \tag{15}$$

Based on these two differences the influence of the increased workload on the QoS of the abiding tenants is expressed as follows.

$$I_{QoS} = \frac{\Delta z_A}{\Delta w} \tag{16}$$

A low value of this metric represents a good isolation as the difference of the QoS in relation to the increased workload is low. Accordingly, a high value of the metric expresses a bad isolation of the system.

Another metric is an enhancement of the previous one, considering the arithmetic mean of I_{QoS} for m disruptive workloads. Whereby the disruptive customers increase their workload equidistant within a lower and upper bound.

$$I_{avg} = \frac{\sum\limits_{i=1}^{m} I_{QoS_m}}{m} \tag{17}$$

3.4 Workload Ratios

The following metrics are not directly associated with the QoS impact resulting from an increased workload of disruptive customers as it was depicted in Figure 3.1. The idea is to compensate the increased workload of disruptive customers and try to keep the QoS for the abiding ones constant by decreasing the workload of the abiding ones (cf. Figure 3.2).

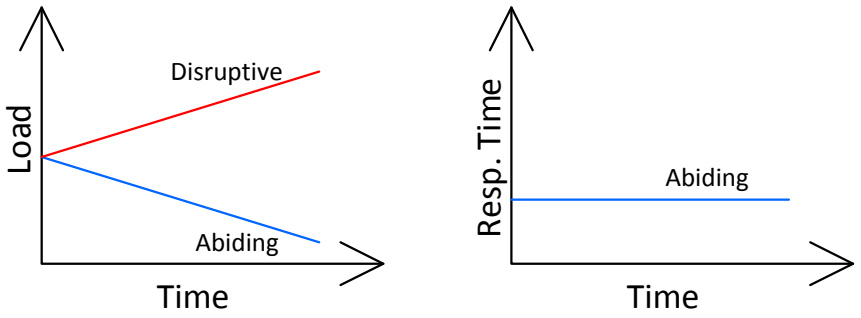


Figure 3.2: Workload Adaption to maintain QoS.

Naturally, this is only possible with the support of the abiding customers and such a behavior does not reflect productive systems. Thus, these metrics are planned to be applied in benchmarks with artificial workloads where a load driver emulates the customers and can be enhanced to follow this behavior.

Assume one starts measuring the isolation behavior of a non-isolated system by continually increasing the disruptive workload W_d. One would expect to observe a decrease of $z_t(W)$ for all customers. In such a situation, $z_t(W)$ would remain unaffected if the workload of the abiding customers W_a is decreased accordingly to compensate for the increase in the disruptive workload. Following this idea, plotting W_a as a result of W_d describes a Pareto optimum of the systems total workload with regards to constant QoS.

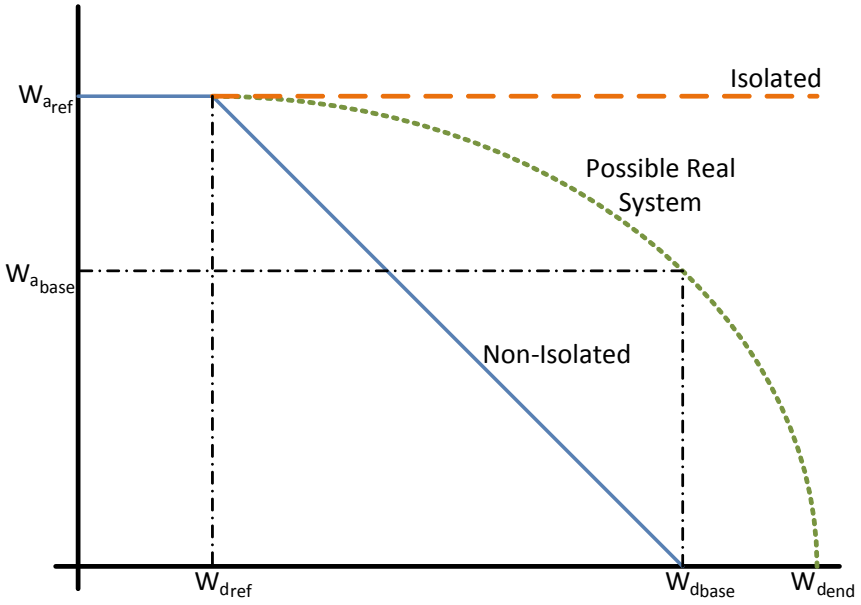


Figure 3.3: Fictitious isolation curve including upper and lower bounds.

In Figure 3.3, the x-axis shows the amount of workload W_d caused by the disruptive tenants, whereas the y-axis shows the amount of the workload W_a caused by the abiding tenants. The blue/solid line shows how W_a has to decrease to maintain the same QoS as in the beginning. In

a non-isolated system this function proportionally decrease linear. For every additional amount added to the disruptive load one has to remove the same amount at the abiding load, because in a non-isolated system the various workload groups would behave if they were one. In a perfectly isolated system the increased W_d has no influence on $z_t(W)$ for all $t \in A$. Thus, W_a would be constant in this case as shown with the red/dashed line in the figure. The red line and the blue line provide exact upper and lower bounds, corresponding to a perfectly isolated and a non-isolated system, respectively. Figure 3.3 shows some important points referenced later and defined in Table 3.2.

Symbol	Definition
W_d	The total workload induced by the disruptive customers: $W_d = \sum_{t \in D} w_t$
$W_{d_{base}}$	The level of the disruptive workload at which the abiding workload in a non-isolated system is decreased to 0 due to SLA violations.
$W_{d_{end}}$	The level of the disruptive workload at which the abiding workload must decreased to 0 in the system under test
$W_{d_{ref}}$	The value of the disruptive workload at the reference point in the system under test. This is the point to which the degree of isolation is quantified. It is defined as the disruptive workload, at which in a non-isolated system the abiding workload begins to decrease.
W_a	The total workload induced by the abiding customers: $W_a = \sum_{t \in A} w_t$
$W_{a_{ref}}$	The value of the abiding workload at the reference point $W_{d_{ref}}$ in the system under test. $W_{a_{ref}} = W_{d_{base}} - W_{d_{ref}}$
$W_{a_{base}}$	The value of the abiding workload corresponding to $W_{d_{base}}$ in the system under test.

Table 3.2: Overview and definition of relevant points.

Based on this approach, several metrics presented in the following. As discussed before, the workload scenarios play an important role, and thus it may be necessary to consider multiple different workload scenarios and average over them.

Significant Points The significant points marked in Figure 3.3 provide several ways to define an isolation metric by themselves. I_{end} is a metric derived by the point at which the workloads of abiding customers have to be decreased to 0 to compensate for the disruptive workload. The metric sets $W_{d_{end}}$ and $W_{a_{ref}}$ in relation. Due to the discussed relationship of the workloads in a non-isolated system and the definition of the various points based on the behavior of such a system the condition $W_{a_{ref}} = W_{d_{base}} - W_{d_{ref}}$ holds. We leverage this relation to simplify our formulas. With Figure 3.3 in mind, I_{end} is defined as follows:

$$I_{end} = \frac{W_{d_{end}} - W_{d_{base}}}{W_{a_{ref}}} \tag{18}$$

Another approach uses $W_{a_{base}}$ as a reference. Setting this value and $W_{a_{ref}}$ in relation results in an isolation metric having a value between $[0, 1]$. The formula for metric I_{base} is below:

$$I_{base} = \frac{W_{a_{base}}}{W_{a_{ref}}} \tag{19}$$

For systems that exhibit a linear degradation of abiding workload, we could also define isolation metrics based on the angle between the observed abiding workloads line segment and the line segment which represents a non-isolated system. However, linear behavior typically cannot be assumed.

Integral Metrics These metrics are based on the area under the curve derived for the measured system $A_{measured}$ set in relation to the area under the curve corresponding to a non-isolated system $A_{nonIsolated}$. The area covered by the curve for a non-isolated system is calculated as $W^2_{a_{ref}}/2$.

The first metric $I_{intBase}$ represents the isolation as the ratio of $A_{measured}$ and $A_{nonIsolated}$ within the interval $[W_{d_{ref}}, W_{d_{base}}]$. Let $f_m : W_d \to W_a$ be a function which returns the residual workload for the abiding customers based on the workload of the disruptive customers. We then define the metric $I_{intBase}$ as follows:

$$I_{intBase} = \frac{\left(\int_{W_{d_{ref}}}^{W_{d_{base}}} f_m(W_d)\,dW_d\right) - W^2_{a_{ref}}/2}{W^2_{a_{ref}}/2} \tag{20}$$

$I_{intBase}$ has a value of 0 in cases the system is not isolated and a value of 1 if the system is perfectly isolated within the interval $[W_{d_{ref}}, W_{d_{base}}]$.

The following metric $I_{intFree}$ allows to use any predefined artificial upper bound p_{end} which represents the highest value of W_d that was measured in the system under test. We define the metric as follows:

$$I_{intFree} = \frac{\left(\int_{W_{d_{ref}}}^{p_{end}} f_m(W_d)\,dW_d\right) - W^2_{a_{ref}}/2}{W_{a_{ref}} \cdot (p_{end} - W_{d_{ref}}) - W^2_{a_{ref}}/2} \tag{21}$$

This metric quantifies the degree of isolation provided by the system for a specified maximum level of injected disruptive workload p_{end}. A value of 1 represents a perfect isolation and a value of 0 a non-isolated system.

3.5 Measurement Methodology

System Setup All isolation metrics are based on the observation of QoS metrics. In Cloud Systems an increasing workload may lead to the allocation of additional resources to keep the observed QoS constant. This behavior belongs to Elasticity 2. To really measure the systems isolation capabilities one has to ensure a constant amount of resources and hardware setup.

Selection of QoS and Workload For a concrete measurement of the particular isolation for one quality metric and workload definition of a system, one has to select an appropriate metric. These metrics have to follow the guidelines discussed at the metrics. Especially one has to consider, that the chosen metric for the workload has to be equivalent for the two groups of users.

Workload Profile For the definition of the workload profile used for measurement we have to consider especially the reference workload profile. The definitions of the upper bounds of the measurements are strongly scenario dependent but might expressed as multiple of the reference workload to make it comparable between various systems. It is conceivable that the observed impact of increasing workload onto the QoS (especially for performance metrics) is rather low when the system is low utilized. Therefore the reference workload should relate to the maximum throughput a system could achieve. As of this point an increasing load of the disruptive customers has a high impact onto the systems performance.

3.6 Discussion

QoS Impact This metric helps system owners to manage their systems, because it indicates the influence of disruptive workload onto the QoS they provide, which is important for capacity planning. QoS-based metrics can prove that a system is perfectly isolated, however they fail in ranking a systems isolation capabilities into the range between isolated and non-isolated. A single I_{QoS} metric can be derived with only two measurements to obtain evidence for one point of increased workload. However, to obtain some more detailed information concerning the systems isolation more measurements are required. Therefore, I_{avg} describes the average isolation value within the upper and lower bound of interest.

Significant Points The metric I_{end} might not be feasible to quantify isolation in well isolated systems. Furthermore, it is not possible to directly deduce relevant system behavior like response times. If the metric is given, it could help to compare two systems regarding the maximum disruptive load they can handle. To determine I_{end}, more measurements as for QoS-based metrics are required.

I_{base} orders a system within the range of isolated and non-isolated systems for one specific point in the diagram. Nevertheless, it does not provide information about the behavior of the system before that point. It is limited to comparing the isolation behavior of the systems at one selected load level and it is inadequate to derive direct QoS-related values. The usefulness of this metric appears to be of limited value in contrast to the integral methods. One advantage is the evidence at a well-defined and reliable point with only two measurements.

Both metrics have some drawbacks resulting from the fact that they do not take into account the curve progression. This means, that in a system which behaves linearly until a short distance from $W_{d_{base}}$ and then suddenly drops to $W_a = 0$, both metrics would have the same value as in the case of a completely non-isolated system which is obviously unfair in this case. Moreover, a well isolated system might require a very high disruptive workload before W_a drops to 0 making it hard to measure the metric in an experimental environment. I_{base} has some further disadvantages given that it is only representative for the behavior of the system within the range of $W_{d_{ref}}$ and $W_{d_{base}}$. Given that the metric does not reflect what happens after $W_{d_{base}}$, it may lead to misleading results for well isolated systems whose respective $W_{d_{end}}$ points might differ significantly.

Integral Metrics $I_{intBase}$ and $I_{intFree}$ are widely comparable metrics. $I_{intBase}$ has the advantage to be measured at a predefined point. For $I_{intFree}$, the endpoint of the interval must be considered as well to have an expressive metric. Both metrics provide good evidence of the isolation within the considered interval, ordered between the magnitudes of isolated and non-isolated systems. They lack in providing information concerning the degree of SLA violation. For example, the SLA violation could be very low and acceptable or critically high in each iteration when we reduce W_a. However, in both cases, the results of the metrics are similar. This limits the value of $I_{intBase}$ and $I_{intFree}$ for system owners/providers. However, for comparison

of systems and analyzing their behavior, the metrics are very useful and can be exploited by developers or researchers. These metrics do not reflect real systems and consequently it is very unlikely to use them outside benchmarking scenarios.

4 Availability

4.1 Goal and Relevance

The goal of this section is to identify critical aspects of measuring Availability of Cloud services, existing metrics and their relation to the de facto standards and conditions that apply to the public Cloud market. Relevance to Cloud environments is high, given that Availability is one of the strong arguments for using Cloud services, together with the elastic nature of the resources and adaptable utilization. Furthermore, availability is a key performance indicator (KPI) included in the majority of public Cloud Service Level Agreements (SLAs).

The final goal is to identify metrics that can be used in various aspects such as provider and service comparisons or potential incorporation into Trust and Reputation mechanisms. Furthermore we intend to highlight aspects that are needed for an Availability benchmark, which in contrast to most cases of benchmarking, it does not refer to the creation of an elementary computational pattern that may be created in order to measure a system's performance but mainly to a daemon-like monitoring client for auditing provider SLAs and extracting the relevant statistics. For this case critical requirements are investigated.

4.2 Prerequisites and Context

Service Level Agreements are contracts between a service provider and a potential client, that govern or guarantee a specific aspect of the offered service. Relevant metrics may include availability, performance aspects, QoS etc. While in research efforts more advanced aspects have been investigated (e.g. direct SLA negotiaton and runtime renegotiation as in Boniface et al. (2010), real-time QoS aspects as in Kousiouris et al. (2011) etc.), in main public Cloud environments they appear as static agreements as in Amazon (2015); Google; Microsoft (2015), prepared in advance by the respective provider, and not available for adaptation to each specific user. Thus a Cloud user by definition accepts these terms when using the specific provider's services, accepting de facto the terms and conditions offered.

4.3 Relevant Metrics Definition

Operational Availability For real time monitoring information regarding services availability, one very interesting approach is CloudSleuth (2015). CloudSleuth has an extensive network of deployed applications in numerous providers and locations and continuously monitors their performance with regard to response times and availability metrics. CloudSleuth's measurement way is not adapted to each provider's SLA definition, so it cannot be used to claim compensation but it relates mainly to the definition of operational availability. Furthermore, it checks the response of a web server (status 200 return type on a GET request). Thus it cannot distinguish between a failure due to an unavailable VM (case of provider liability) or due to an application server crash (customer liability in case of IaaS deployment, provider liability in case of PaaS) or pure application fault (customer liability). On the other hand, it follows a normal availability definition that makes it feasible to compare services from different providers, a process which cannot be performed while following their specific SLA definitions, since they have differences in the way they define availability.

$$Availability = \frac{TotalSamples - UnavailableSamples}{TotalSamples} \tag{22}$$

CloudSleuth Formula for Availability

In Chandler et al. (2012), availability is defined in relation to the Mean Time Between Failures metric, so that the metric avoids cases where small uptime is hidden from the fact of

very low downtime. While this is reasonable, commercial Cloud providers tend to consider as uptime the entire month duration (as mentioned in the following chapters) rather than only the actual uptime the services were used by the end user. A very thorough monitoring tools analysis is conducted in Aceto et al. (2013). While numerous tools exist for focusing on service availability (and other non functional properties), it is questionable whether the way they are calculating the specific metrics is compliant to the relevant definitions in commercial public Clouds and their respective SLAs. In Li et al. (2012a), an investigation of new metrics and relevant measurement tools expands across different areas such as network, computation, memory and storage. However, availability is again not considered against the actual SLA definitions.

De Facto Industrial SLAs Examination As mentioned in Section 4.2, commercial providers follow a considerably different approach, defining by themselves the metric of availability that is considered in their respective SLAs. In many cases this implies a different calculation based on the same samples and a set of preconditions that are necessary for a specific service deployment to be included under the offered SLA. In order to investigate the ability to provide an abstraction layer for this type of services and thus create a generic and abstracted benchmarking client, one needs to identify common concepts in the way the SLAs are defined for the various providers (we considered Amazon EC2, Google Compute and Windows Azure). The first and foremost point of attention is the generic definition of availability, which is included in Equation 23.

$$Availability = \frac{TotalMinutesOfMonth - DowntimeMinutes}{TotalMinutesOfMonth} = \frac{TotalMinutesOfMonth - \sum DownTimePeriods_i}{TotalMinutesOfMonth} \tag{23}$$

Provider based definition of availability

This may seem the same as the one defined in Equation 22, however the providers typically define the downtime period as a step function (x in minutes):

(24)

$$DownTimePeriod(x) = \begin{cases} 0 \text{ if } x < minimum \text{ (Google: 5, Amazon: 1, Azure: 1)[min]} \\ x \text{ if } x > minimum \end{cases}$$

Furthermore, one other key point is the existence of preconditions needed for an SLA to be applicable for a specific group of services used (e.g. need to have multiple VMs deployed in different availability zones). There are variations between provider definitions with regard to this aspect, with relation to where the VMs are deployed or if further action is necessary after the realization that a virtual resource is unavailable (e.g. restart of the resource). In a nutshell, the similarities and differences of the three examined providers are the following:

1. Common concepts
   (a) Quantum of downtime period: providers do not accept that a downtime period is valid, unless it is higher than a specific quantum of time.
   (b) Discount formats as compensation for downtime with relation to the monthly bill.
   (c) Calculation cycle of 1 month.
   (d) More than one VM to accept SLA applicability.
   (e) VM distribution across at least 2 availability zones.

(f) Simultaneous unavailability for all deployed VMs for an overall sample to be considered as unavailable.

2. Differences

(a) Quantum size. This is considered minor since the format is the same.

(b) Number of discount levels. The three providers offer different discounts for various levels of deviation from the availability goal. This is considered minor since the format is the same.

(c) For the Azure Compute case, it seems that more than one instances for the same template ID must be deployed. However, it seems also that the overall time interval refers only to the time a VM was actually active.

(d) Restart from template is necessary in Google App Engine before validating an unavailable sample

4.4 Abstracted and Comparable Metrics

Given the differences in the aforementioned availability definitions, it is not feasible to directly compare provider-defined availability metrics between providers, since these differ in definition. For this reason, it is more meaningful to either follow a more generic direct definition as done in CloudSleuth (2015) or abstract to a more generic concept which is the SLA adherence level. This can be defined as the ratio of violated SLAs over the overall examined SLAs. Since SLA period is set to monthly cycles, this may be the minimum granularity of observation.

SLA Adherence Levels

$$SLA\_Adherence = \frac{violatedSLAs}{overallObservedSLAs} \tag{25}$$

SLA Adherence Metric

Special attention must be given for cases that sampling is not continuous, indicating that the client did not have running services for a given period, applicable for an SLA. These cases must be removed from such a ratio, especially for the cases that no violations are examined in the limited sampling period, given that no actual testing has been performed. If on the other hand even for a limited period a violation is observed, then this may be included. Furthermore, SLA adherence may be grouped according to the complexity of the examined service, as mentioned in more detail in Section 1.5.2.

SLA Strictness Levels Besides SLA adherence, other metrics may be defined in order to depict the strictness of an SLA. As a prerequisite, we assume that the metric must follow a "higher is stricter" approach. Stricter implies that it is more difficult for a provider to maintain this SLA. In order to define such a metric initially one needs to identify what are the critical factors that may affect strictness levels. Factors should be normalized to a specific interval (e.g. 0 to 1) and appropriate levels for them may be defined. Indicative factors may include:

1. Size of the minimum continuous downtime period (Quantum of downtime period q). A higher size means that the SLA is more relaxed, giving the ability to the provider to hide outages if they are less than the defined interval. The effect of such a factor may be of a linear fashion (e.g. 1-q). Necessary edges of the original interval (before the normalization process) may be defined per case, based e.g. on existing examples of SLAs.

2. Ability to use the running time of the services and not the overall monthly time, denoted by a Boolean variable t. This would be stricter in the sense that we are not considering the time the service is not running as available. The effect of such a factor may be of a Boolean fashion (0 false, 1 true)

3. Percentage of availability that is guaranteed. Again this may be of a linear fashion, with logical intervals defined by the examined SLAs.

4. Existence of performance metrics (e.g. response timing constraints). This may be a boolean feature x, however its values may be set to higher levels (0 or 5). The importance of this will be explained briefly.

The added value of such a metric may be in the case we have to deploy applications with potentially different characteristics and requirements (as one would expect). For example, having *soft real-time applications* would imply that we definitely need to have feature 4. Other less demanding applications may be accommodated by services whose SLAs are less strict. Thus suitable **value intervals** may be adjusted for each feature. If we use a value of 5 for the true case of feature 4, and all the other features are linked in such a manner that their accumulative score is not higher than 5, then by indicating a necessary strictness level of 5 implies on a numerical level that feature 4 needs definitely to be existent in the selected Cloud service.

Depending on the application types and their requirements and based on the metric definition, one can define categories of strictness based on the metric values that correspond to according levels (e.g. medium strictness needs a score from 2 to 3 etc.). It is evident that such a metric is based only on the SLA analysis and is static, if the SLA definition is not changed. The indicative formula for the case of equal importance to all parameters appears in Equation 25.

$$S = t + (1 - s_1 q) + s_2 p + x \text{ where}$$

s_i : normalization factor for the continuous variables so that (s_1*q) $\in [0,1]$ and (s_2*p) $\in [0,1]$

$t \in \{0,1\},\ x \in \{0,1\}$

(26)

SLA Strictness definition formula

For the normalization intervals, for p we have used 99% and 100% as the edges, given that these were the ranges encountered in the examined SLAs. For q we have used 0 and 10 minutes as the edges. 0 indicates the case where no minimum interval is defined (thus abiding by the formula in Equation 22) and 10 the maximum interval in examined Compute level SLAs. However there are larger intervals (e.g. 60 minutes) in terms of other layer SLAs (Azure Storage). The limit to 60 has been tried out in the q' case that is included in Table 1, along with the example of the other factors and the overall values of the SLA strictness metrics in the 3 examined public SLAs.

Table 1: Indicative application of the SLA Strictness metric on existing public Cloud SLAs

Provider/Service	t	q	q'	p	x	S	S'
Google Compute	0	5 (norm:0.5)	5 (norm: 0.0833)	99.95 (norm:0.5)	0	1	1.4167
Amazon EC2	0	1(norm: 0.1)	1(norm: 0.0167)	99.95 (norm:0.5)	0	1.4	1.4833
Microsoft Azure	1	1(norm: 0.1)	1(norm: 0.0167)	99.95 (norm:0.5)	0	2.4	2.4833

An example of x not being 0 would be the Azure Storage SLA, where unavailability is also determined by response time limits to a variety of service calls.

4.5 Measurement Methodology

In order to create an Availability Benchmark for the aforementioned SLAs, the following requirements/steps in the methodology must be undertaken:

1. Complete alignment to each provider definition in order to achieve **Non-repudiation**, including the following:
   (a) Necessary **preconditions** checking in terms of number and type
   (b) Availability **calculation** formula
   (c) **Dynamic** consideration of user actions (e.g. starting/stopping of a VM) that may influence SLA applicability
   (d) Downtime due to **maintenance**
2. General assurance mechanisms in terms of faults that may be accredited to 3^{rd} party services/equipment
   (a) For example testing of general Internet connectivity on the client side
   (b) Front-end API availability of provider
   (c) Monitoring daemon not running for an interval at the client side. This can be covered by appropriate assumptions, e.g. if logs are not available for an interval then the services are considered as available
3. Logging mechanisms that persist the necessary range and type of information

System Setup System setup should include a running Cloud service. Furthermore, it should include the benchmark/auditing code (according to the aforementioned requirements taken under consideration) that is typically running externally to the Cloud infrastructure, in order to cover the case that connectivity exists internally in the provider but not towards the outside world.

Workload Given that this is a daemon-like benchmark, the concept of workload is not directly applicable. The only aspect of workload that would apply would be for the number of Cloud service instances to be monitored and the only constraint is that these cover the preconditions of the SLA. However an interesting consideration in this case may be the differentiation based on the complexity of the observed service (in terms of number of virtual resources used), given that this would influence the probabilities of not having a violation.

If we consider the case of a typical Cloud deployment at the IaaS level, we may use N availability zones (AZ), in which M virtual machines are deployed. An availability zone is typically defined as a part of the data center that shares the same networking and/or power supply. Thus the usage of multiple AZs eliminates the existence of a single point of failure for the aforementioned risks. In the generic case, M is a 1xN vector, containing the number of VMs deployed in each AZ. For simplicity purposes we assume that M is the same across all AZs. If P_{POW} is the probability of power supply failure and P_{AZNET} the probability of network failure, P_{PH} the risk of the physical host in which a VM is running to fail and P_{VM} the risk of the VM to fail then, assuming that these probabilities are mutually exclusive, depending on different factors, the overall probability of failure for a deployment in one AZ is given by Equation 24. The service is deemed as unavailable in one specific AZ if power or network connectivity is lost across the AZ or if all VMs in that AZ are at the same time unavailable.

$$\begin{aligned} P_{OTH} &= P_{POW} + P_{AZNET} \\ P_{NODE} &= P_{PH} + P_{VM} \\ P_{AZ} &= P_{OTH} + \prod_{i=1}^{M} P_{NODEi} = P_{OTH} + P_{NODE}^{M} \end{aligned} \tag{27}$$

Overall probability for an AZ to fail

If M VMs are deployed in each one of the N AZs, then the overall failure probability, assuming that the various AZs have similar power and network probability failures and given that we are not aware of the affinity of VM placement across physical nodes thus we can assume that different physical hosts are used for each VM, we have the overall failure (and overall unavailability) probability to be given by Equation 25.

$$P_{OVERALL} = (P_{OTH} + P_{NODE}^{M})^{N} = \sum_{k=0}^{N} \frac{N!}{K!(N-K)!} P_{OTH}^{K} P_{NODE}^{M(N-K)} \tag{28}$$

Overall service unavailability probability

Thus the significant factors that indicate the complexity (M and N) can be used as a generic metric of "workload". Furthermore they can be used to classify results of Equation 24 in categories according to the service complexity.

4.6 Discussion

While availability has been defined in the literature in various ways, existing mainstream public Clouds such as Amazon, Google and Azure have separate definitions, which may be similar but not identical even to each other. Thus direct comparison of providers based on these metrics can not be achieved and especially benchmarked against the guarantees they advertise in their respective SLAs. In this section, an analysis is performed regarding the similarities of provider definitions and how these can lead to guidelines regarding the implementation of benchmarking clients for identifying provider fulfillment of the issued Service Level Agreements towards their customers.

Furthermore, we define a simple yet directly comparable (between providers) metric of SLA adherence. Classes of workload can be identified based on the size and deployment characteristics of the measured service thus further refining the aforementioned comparison.

5 Operational Risk

5.1 Goal and Relevance

Cloud services are commonly inscribed with performance guarantees which play important role on cloud resource-management. These performance guarantees are actually levels of performance described in Service-Level-Agreements (SLA) having been agreed upon between cloud providers and customers. The violation of SLAs impose financial penalties on cloud providers as well as reputation cost in cloud market. Therefore, a metric providing the assessment of service performance-levels and of possible SLA violations is needed. Such performance levels can be characterized as expected levels for systems in order to perform without implications. Another definition of the expected system performance-levels may be the performance of common systems whose performance functions as baseline factor to assess other systems. Considering the performance expectations of systems, the *operational risk* is defined as the likelihood of service performance to be on expected levels.

The operational risk metric complements the purpose of the other metrics proposed in previous sections. Risk management in cloud computing reflects the necessity of knowing the severity of changes in cloud infrastructures by evaluating the implications. While metrics specialized for particular cloud features (e.g., Elasticity, Isolation and Availability) measure the respective system performance regarding the features, the role of operational risk is different. The operational risk can be measured for any of the mentioned features by defining their type of performance and their respective expected levels. In other words, an operational risk metric can reflect the risk that the system performance deviates from the expected performance levels despite the type of performance.

Having a representative metric of operational risk contributes to a higher level approach towards the performance evaluation of complex systems as the cloud ecosystems. In addition, the metric provides a level of abstraction of the overall system performance to stakeholders who concern about metrics depicting levels of system risk.

5.2 Prerequisites and Context

In this section, we describe the operational risk of cloud services deriving from their performance levels when running in cloud infrastructures. The term *risk* is also used in literature for describing security issues of cloud but the notion of security is out of scope of the presented metric.

The *service performance* in clouds refers to multiple kinds of performance that can be measured in cloud services. A possible distinction among these performance types is the level of service (e.g., IaaS, PaaS, SaaS) in cloud system. The service level indicates the service performance for that level and also the suitable metrics measuring the performance . For example, the *response-time* is useful for measuring service performance in PaaS, SaaS levels while the performance on leveraging service resources (e.g., CPU, memory) is addressed by IaaS-oriented metrics. In this section, we focus on the service performance in the level of IaaS, thus the operational risk utilizes IaaS-level metrics and refers to performance levels which reflect the resource utilization by cloud services.

5.3 Proposed Operational Risk Metrics

The idea behind the *operational risk* metric proposed in this section is to quantify the variation of service performance between the service running in a dedicated environment and the performance when the service running in another, non-dedicated environment. The former provides an isolated environment where resources are always available to the service under test. In contrast, due to concurrent operations of services in a non-dedicated environment (i.e., common

cloud system), services share the resources resulting in performance interference among the services Zhuravlev et al. (2010). This interference affects the performance delivered in services and in turn, this performance may deviate from the service performance in the dedicated environments. This deviation directly impacts the service performance resulting in its degradation.

For this reason, the term *risk* is used to imply the likelihood that the actual service performance in the non-dedicated cloud will deviate from the demanded performance a service requires, which is reflected by the performance in the dedicated system.

Related Metrics

The risk metric reveals an additional aspect of the service performance in the cloud which is the degree to which service performance is degraded. Figure 5.1 shows the three metrics referring to resource type of a cloud service. The current usage (U) of the resource, the demanded (D) amount the service requires and the provisioned (P) amount of resource which is the upper limit of resource the service is able to consume and is given by the resource manager.

As Figure 5.1 shows, D line can be higher than the U line as a service may not receive what it requires due to contention issues in the system, e.g., resource overcommitment. We consider that usage U cannot be higher than the demanded amount D ($U \not> D$) because the real usage of resource cannot exceed the requirements of service in resources. Similarly, $U \not> P$ as the actual resource consumption can only reach up to the resource limit that has been set by the resource manager.

Relative Levels of Metrics Considering the three mentioned metrics, we show the possible cases about the relative levels of the metrics. In an ideally auto-scaling and isolated environment, the service is provisioned and consumes the demanded amount of resource ($P = D = U$), as in period T_2 in Figure 5.1. The resource manager provides the demanded resources ($P = D$) and the service utilizes all the demanded resources ($D = U$) without interfering with other co-hosted services.

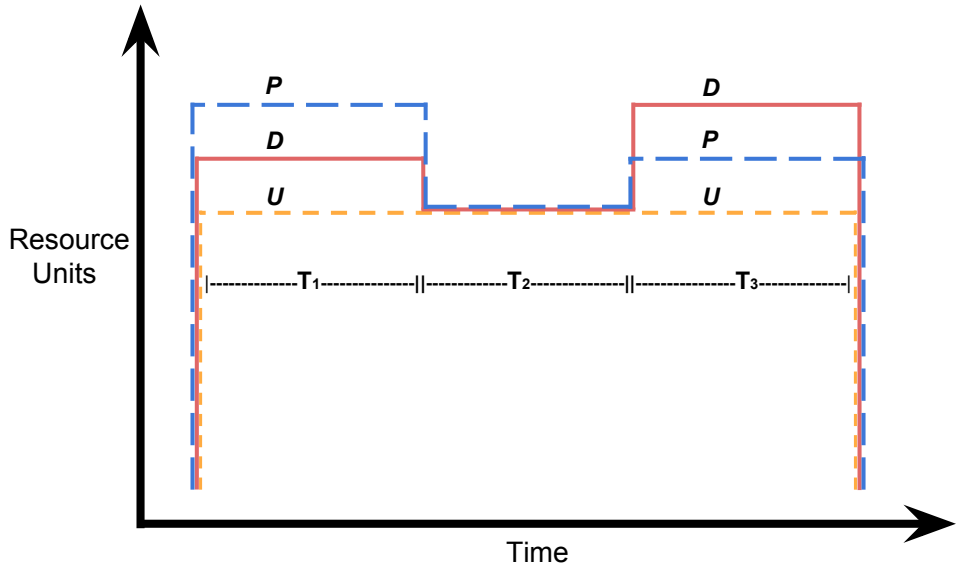


Figure 5.1: Metrics composing the Operational Risk metric for cloud service.

However, the case of the three coinciding lines is not applicable in real environments. An auto-scaling mechanism is not perfectly accurate and creates incompatibilities between the pro-

visioned and the demanded resources. When a service demands fewer resources than it has been provisioned ($D < P$), then the system hosts an over-provisioned service. In the opposite way, the system has under-provisioned resources ($D > P$) to service. The two mentioned cases are depicted in periods T_1, T_3 in Figure 5.1 respectively.

Another possible case of the metric levels is the relative position between D and U. As mentioned before, the real usage cannot be higher than the demanded usage, however coinciding lines of U and D is not always the case due to contention issues in the system. We call the deviation between D and U, which is always non-negative ($D \geq U$), as *contention-metric* and it presents the gap between the demanded amount of resource and the real consumption of resource at specific time. The higher the value of the contention-metric, the more severe is the resource-contention which is experienced by the service; hence degrading the service performance. For instance, in Figure 5.1, the service experiences more contention in its resources during period T_3 than during period T_1 as the gap between the resource lines D, U is bigger in the former period.

Definition of Metric

The operational risk metric presented in this Section incorporates the three resource metrics mentioned before and splits the metric functionality into two partial metrics, the *provision risk* (r_p) and the *contention risk* (r_c). The objective of the two partial metrics is the combination of two cloud features, elasticity and isolation. This aggregation will benefit the metric users to have more comprehensive view of the system performance and adjust performance issues towards their needs.

Provision Risk We define as risk of provisioned resources or *provision risk* (r_p), the degree to which the demanded resources meet the provisioned resources.

$$r_p = \frac{1}{T} \int_T \frac{P_t - D_t}{P_t} \, dt, \qquad [-1, 1] \tag{29}$$

where P_t and D_t are the provisioned and the demanded resources respectively at time t. The T value is the time period which we measure the two metrics. The integral value of r_p measures the relative squashed area enclosed by the resource values of P and D for time period T. The value of r_p ranges between $[-1, 1]$ indicating an under-provisioning situation when $r_p \in [-1, 0)$ and an over-provisioning case when $r_p \in (0, 1]$. The zero value indicates an accurate provision of resources according to the demanded resources. Thus, the closer the value of r_p is to zero, the less risk is indicated for the service.

Contention Risk Similarly, we define the *contention risk* (r_c). This metric utilizes the resource values of demanded and used resources over time T. The metrics D_t and U_t indicate the values of the respective metrics at time t.

$$r_c = \frac{1}{T} \int_T \frac{D_t - U_t}{D_t} \, dt, \qquad [0, 1] \tag{30}$$

The r_c value is non-negative as the amount of used resources cannot exceed the amount of demanded resources. The higher the value of r_c, the more risk is estimated for a service to not receive the demanded performance due to resource contention.

Service Risk The risk for a cloud service should be composed by the combination of the two aforementioned risk metrics in order to have better overview on the service status regarding risk levels.

We define the *risk of service* (r_e) as the degree to which a cloud service performs as expected. The expected performance derives from the expected performance of an elastic system and the expected performance with respect to the experienced contention in a service. Both performance levels contribute to evaluate if a system supplies the service adequately with enough resources.

$$r_e = w_p \times |r_p| + w_c \times r_c = \frac{1}{T}\left(w_p \times \int_T \frac{|P_t - D_t|}{P_t}\,dt + w_c \times \int_T \frac{D_t - U_t}{D_t}\,dt \right), \qquad [0,1] \qquad (31)$$

with $w_p, w_c \in [0,1], w_p + w_c = 1.$

For the metric r_e, the difference between P_t and D_t is an absolute value because we do not focus on the provisioning type of risk but only for the level of severity that the difference between the two metrics reflects. Factors $w_p, w_c \in [0,1]$ are used to weight the operational risk value according to user needs.

System Risk The operational risk of cloud system should be an overview of the corresponding risk values of the services in the system. The aggregation method of service risks, which calculates the system-risk value, deviates according to user needs. The method will probably be a variability metric which combines values according to different purposes. For example, cloud providers who want to test the risk levels of the system may use quantile values to depict the risk level of some service groups. For cloud customers, who want to deploy cloud services into an environment with stable risk levels, a variability metric like the interquartile range (IQR) will show the dispersion of service risks in the system and thus, the performance variability of the system.

5.4 Measurement Methodology

To measure effectively the operational risk of cloud systems, we have to determine which monitored data are the respective metrics that operational risk is built upon. Additionally, the resource type has to be decided in order the appropriate resource metrics to be declared. We focus on the most common resource types that are currently available to clouds, i.e., CPU, memory, network and storage.

Metrics P & U The metric definitions of provision (P) and usage (U) are straight-forward and one can easily monitor the respective values. Metric P refers to the capacity that resource manager has provisioned to service and metric U is the actual capacity that service uses. In any resource type, the corresponding metrics of P and U can be readily monitored.

Metric D Although it may be confusing how the demanded amount of the aforementioned resource types is estimated, there is currently enough research on that topic. For the resource type of CPU, prediction models have been introduced in Isci et al. (2010) and contemporary cloud monitors provide the metric of CPU_Demand. For the memory resource, the demanded amount can also be estimated as in Zhou et al. (2004) and used as the possible memory capacity that service needs at specific time. For the resource types of network and storage, the metric D is simpler to be calculated as the demanded capacities are either the size or the number of requests of that resource received by the resource manager. The resource manager, after collecting the requests, handles a subset of these requests (i.e., metric U) due to the system load.

Weights w_p, w_c The values of the two weights w_p, w_c represent the importance of the two metrics r_p, r_c respectively. One has to consider the purpose of benchmarking the cloud system in order to define similarly the weights. The *contention risk* may be more important for testing the impact of co-location of cloud services ($w_p < w_c$) whereas the *provision risk* represents better the operational risk when selecting the most promising elastic policy for a system.

5.5 Related Work

Risk management and resource contention are not new subjects in cloud research. Risk management has been defined as decision paradigm in Grids Djemame et al. (2006) where selection among multiple infrastructures should have been taken into account to host an application. In this work and also in the extended work of Ferrer et al. (2012), historical records about SLA violations are used to assess the system risk for an incoming application to fulfill the agreed objectives. The risk levels are also evaluated according to the provisioned mismatches of the elasticity mechanism. Our approach of operational risk considers the cloud feature of elasticity and incorporates it with the risk of performance interference of services.

In Tang et al. (2011), the authors investigate the interference of services in memory. The presented results on performance degradation use as baseline-level the performance of a service running alone in the system under test. In Zhuravlev et al. (2010), a similar approach measuring the performance degradation is used with a more explicit definition of the degradation. The authors define the relative performance degradation of memory components according to the performance level of a running-alone service. We extend this related work considering the impact of elasticity in the system. The degradation of performance due to resource contention is affected by the elastic mechanisms in clouds and this emerges the necessity of incorporating the two notions.

5.6 Discussion

In this section, we present the feature of operational risk in clouds and introduce a representative metric evaluating the risk in cloud services and systems not to perform as expected.

The notion of risk differentiates from the elasticity feature in cloud because elasticity takes into account the provisioned and the demanded values of service resource to cope with the service load. In contrast, both performance isolation and availability utilize demand and usage metrics for different reasons. Performance isolation concerns about the contention that a service may experience and evaluates the severity of the interference among services while availability evaluates the periods where demand is present but the usage of resources cannot be achieved.

The operational risk incorporates the three resource-level metrics and complements the other metrics described in previous sections in order to assess the severity of performance degradation regarding the mentioned cloud features.

Usability

The operational risk metric can be used as evaluation of cloud services and systems to assess whether the performance guarantees are met.

The measurement of service-performance in the cloud is important for both customer and cloud provider. The customer wants to maximize profit by delivering good QoS to clients. Therefore, the service performance is utilized by the customer to check the progress of service-runtime as well as to compare and select the cloud environment which meets the service needs the most.

For cloud providers, they are interested in the levels of service performance because they are burdened with financial fees when SLA violations occur. Moreover, there is a reputation cost

for cloud providers when not delivering good performance results to customers, thus, providers also utilize service-performance metrics to maintain competition in the cloud market.

6 Conclusion

Because cloud computing services, and the systems underlying them, already account for a large fraction of the information and communication technology (ICT) market, understanding their non-functional properties is increasingly important. Building for this tasks cloud benchmarks could lead to better system design, tuning, and selection. Responding to this need, in this report, we highlight the relevance of new non-functional system properties emerging in the new context of cloud computing, namely elasticity, performance isolation, availability and operational risk. We discuss these properties in depth and select existing or propose new metrics that are capable to quantify these properties. Thus, for these four properties we lay a foundation for benchmarking cloud computing settings.

As future activities, we plan to conduct real-world experiments that underline the applicability and usefulness of the proposed metrics, also refining the corresponding measurement approaches. As a next step, we are working on an extensive review of existing cloud-relevant benchmarks and connected domains like big data, web services, and graph processing.

References

[acm 2012] The 2012 ACM Computing Classification System. http://www.acm.org/about/class/2012, 2012. [Online: accessed January 2016].

[Aceto et al. 2013] G. Aceto, A. Botta, W. de Donato, and A. Pescapa. Cloud monitoring: A survey. *Computer Networks*, 57(9):2093 – 2115, 2013. ISSN 1389-1286. doi: http://dx.doi.org/10.1016/j.comnet.2013.04.001. URL http://www.sciencedirect.com/science/article/pii/S1389128613001084.

[Almeida et al. 2013] R. F. Almeida, F. R. Sousa, S. Lifschitz, and J. C. Machado. On Defining Metrics for Elasticity of Cloud Databases. In *Proceedings of the 28th Brazilian Symposium on Databases*, 2013. URL http://sbbd2013.cin.ufpe.br/Proceedings/artigos/sbbd_shp_12.html.

[Amazon 2015] Amazon. EC2 Compute SLA, 2015. URL http://aws.amazon.com/ec2/sla/.

[Binnig et al. 2009] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing. How is the Weather Tomorrow?: Towards a Benchmark for the Cloud. In *Proceedings of the Second International Workshop on Testing Database Systems*, DBTest '09, pages 9:1–9:6, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-706-6. doi: 10.1145/1594156.1594168. URL http://doi.acm.org/10.1145/1594156.1594168.

[Boniface et al. 2010] M. Boniface, B. Nasser, J. Papay, S. Phillips, A. Servin, X. Yang, Z. Zlatev, S. Gogouvitis, G. Katsaros, K. Konstanteli, G. Kousiouris, A. Menychtas, and D. Kyriazis. Platform-as-a-service architecture for real-time quality of service management in clouds. In *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*, pages 155–160, May 2010. doi: 10.1109/ICIW.2010.91.

[Chandler et al. 2012] D. Chandler, N. Coskun, S. Baset, E. Nahum, S. R. M. Khandker, T. Daly, N. W. I. Paul, L. Barton, M. Wagner, R. Hariharan, and Y. seng Chao. Report on Cloud Computing to the OSG Steering Committee. Technical report, Apr. 2012. URL http://www.spec.org/osgcloud/docs/osgcloudwgreport20120410.pdf.

[CloudSleuth 2015] CloudSleuth. Cloudsleuth monitoring network, 2015. URL https://cloudsleuth.net/.

[Cooper et al. 2010] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with YCSB. In *Proceedings of the 1st ACM symposium on Cloud computing*, SoCC '10, pages 143–154, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0036-0. doi: 10.1145/1807128.1807152. URL http://doi.acm.org/10.1145/1807128.1807152.

[Djemame et al. 2006] K. Djemame, I. Gourlay, J. Padgett, G. Birkenheuer, M. Hovestadt, O. Kao, and K. Voß. Introducing Risk Management into the Grid. *The 2nd IEEE International Conference on eScience and Grid Computing eScience2006*, 2006.

[Dory et al. 2011] T. Dory, B. Mejías, P. V. Roy, and N.-L. Tran. Measuring Elasticity for Cloud Databases. In *Proceedings of the The Second International Conference on Cloud Computing, GRIDs, and Virtualization*, 2011. URL http://www.info.ucl.ac.be/~pvr/CC2011elasticityCRfinal.pdf.

[Duboc 2009] L. Duboc. *A Framework for the Characterization and Analysis of Software Systems Scalability.* PhD thesis, Department of Computer Science, University College London, 2009. http://discovery.ucl.ac.uk/19413/1/19413.pdf.

[Duboc et al. 2007] L. Duboc, D. Rosenblum, and T. Wicks. A Framework for Characterization and Analysis of Software System Scalability. In *Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC-FSE '07)*, pages 375–384. ACM, 2007. doi: http://doi.acm.org/10.1145/1287624.1287679.

[European Commission 2014] European Commission. Uptake of cloud in europe. Final Report. Digital Agenda for Europe report. Publications Office of the European Union, Luxembourg, 2014.

[Ferrer et al. 2012] A. J. Ferrer, F. Hernández, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. K. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgó, T. Sharif, and C. Sheridan. OPTIMIS: A holistic approach to cloud service provisioning. *Future Generation Computer Systems*, 2012.

[Fleming and Wallace 1986] P. J. Fleming and J. J. Wallace. How Not to Lie with Statistics: The Correct Way to Summarize Benchmark Results. *Commun. ACM*, 29(3):218–221, Mar. 1986. ISSN 0001-0782. doi: 10.1145/5666.5673. URL http://doi.acm.org/10.1145/5666.5673.

[Folkerts et al. 2012] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun. Benchmarking in the Cloud: What It Should, Can, and Cannot Be. In R. Nambiar and M. Poess, editors, *Selected Topics in Performance Evaluation and Benchmarking*, volume 7755 of *Lecture Notes in Computer Science*, pages 173–188. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-36726-7. doi: 10.1007/978-3-642-36727-4_12. URL http://dx.doi.org/10.1007/978-3-642-36727-4_12.

[Google] Google. Compute Level SLA. URL https://cloud.google.com/compute/sla.

[Herbst et al. 2013] N. R. Herbst, S. Kounev, and R. Reussner. Elasticity in Cloud Computing: What it is, and What it is Not (Short Paper). In *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013)*. USENIX, June 2013. URL https://www.usenix.org/conference/icac13/elasticity-cloud-computing-what-it-and-what-it-not.

[Herbst et al. 2015] N. R. Herbst, S. Kounev, A. Weber, and H. Groenda. Bungee: An elasticity benchmark for self-adaptive iaas cloud environments. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '15, pages 46–56, Piscataway, NJ, USA, 2015. IEEE Press. URL http://dl.acm.org/citation.cfm?id=2821357.2821366.

[Huber et al. 2011] N. Huber, M. von Quast, M. Hauck, and S. Kounev. Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments. In *Proceedings of the 1st International Conference on Cloud Computing and Services Science (CLOSER 2011)*, pages 563 – 573. SciTePress, May 2011. ISBN 978-989-8425-52-2.

[Huppler 2009] K. Huppler. Performance Evaluation and Benchmarking. chapter The Art of Building a Good Benchmark, pages 18–30. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-10423-7. doi: 10.1007/978-3-642-10424-4_3. URL http://dx.doi.org/10.1007/978-3-642-10424-4_3.

[Huppler 2012] K. Huppler. Benchmarking with Your Head in the Cloud. In R. Nambiar and M. Poess, editors, *Topics in Performance Evaluation, Measurement and Characterization*, volume 7144 of *Lecture Notes in Computer Science*, pages 97–110. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-32626-4. doi: 10.1007/978-3-642-32627-1_7. URL http://dx.doi.org/10.1007/978-3-642-32627-1_7.

[IDC 2013] IDC. Worldwide and regional public it cloud services: 2013-2017 forecast. IDC Tech Report. [Online] Available: www.idc.com/getdoc.jsp?containerId=251730, 2013.

[Iosup et al. 2011] A. Iosup, N. Yigitbasi, and D. H. J. Epema. On the performance variability of production cloud services, 2011. URL http://dx.doi.org/10.1109/CCGrid.2011.22.

[Isci et al. 2010] C. Isci, J. Hanson, I. Whalley, M. Steinder, and J. Kephart. Runtime demand estimation for effective dynamic resource management. pages 381–388, April 2010. ISSN 1542-1201. doi: 10.1109/NOMS.2010.5488495.

[Islam et al. 2012] S. Islam, K. Lee, A. Fekete, and A. Liu. How a Consumer Can Measure Elasticity for Cloud Platforms. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, ICPE '12, pages 85–96, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1202-8. doi: 10.1145/2188286.2188301. URL http://doi.acm.org/10.1145/2188286.2188301.

[Jogalekar and Woodside 2000] P. Jogalekar and M. Woodside. Evaluating the scalability of distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 11:589–603, 2000.

[Kousiouris et al. 2011] G. Kousiouris, D. Kyriazis, S. Gogouvitis, A. Menychtas, K. Konstanteli, and T. Varvarigou. Translation of application-level terms to resource-level attributes across the cloud stack layers. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 153–160, June 2011. doi: 10.1109/ISCC.2011.5984009.

[Krebs et al. 2012] R. Krebs, C. Momm, and S. Kounev. Metrics and techniques for quantifying performance isolation in cloud environments. In *Proceedings of the 8th international ACM SIGSOFT conference on Quality of Software Architectures*, QoSA '12, pages 91–100, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1346-9. doi: 10.1145/2304696.2304713. URL http://doi.acm.org/10.1145/2304696.2304713.

[Krebs et al. 2014] R. Krebs, C. Momm, and S. Kounev. Metrics and Techniques for Quantifying Performance Isolation in Cloud Environments. *Elsevier Science of Computer Programming Journal (SciCo)*, Vol. 90, Part B:116–134, 2014.

[Kuperberg et al. 2011] M. Kuperberg, N. R. Herbst, J. G. von Kistowski, and R. Reussner. Defining and Quantifying Elasticity of Resources in Cloud Computing and Scalable Platforms. Technical report, Informatics Innovation Center at Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, 2011. http://digbib.ubka.uni-karlsruhe.de/volltexte/1000023476.

[Li et al. 2010] A. Li, X. Yang, S. Kandula, and M. Zhang. CloudCmp: Comparing Public Cloud Providers. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*,

IMC '10, pages 1–14, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0483-2. doi: 10.1145/1879141.1879143. URL http://doi.acm.org/10.1145/1879141.1879143.

[Li et al. 2012a] Z. Li, L. O'Brien, H. Zhang, and R. Cai. On a catalogue of metrics for evaluating commercial cloud services. In *Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on*, pages 164–173, Sept 2012a. doi: 10.1109/Grid.2012.15.

[Li et al. 2012b] Z. Li, L. O'Brien, H. Zhang, and R. Cai. On a Catalogue of Metrics for Evaluating Commercial Cloud Services. In *Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on*, pages 164–173, Sept 2012b. doi: 10.1109/Grid.2012.15. URL http://dx.doi.org/10.1109/Grid.2012.15.

[Microsoft 2015] Microsoft. Azure compute level sla, 2015. URL http://www.microsoft.com/en-us/download/details.aspx?id=38427.

[Plummer et al. 2009] D. C. Plummer, D. M. Smith, T. J. Bittman, D. W. Cearley, D. J. Cappuccio, D. Scott, R. Kumar, and B. Robertson. Study: Five Refining Attributes of Public and Private Cloud Computing. Technical report, Gartner, 2009. URL http://www.gartner.com/newsroom/id/1035013. http://www.gartner.com/DisplayDocument?doc_cd=167182, last consulted Feb. 2013.

[Shawky and Ali 2012] D. Shawky and A. Ali. Defining a Measure of Cloud Computing Elasticity. In *Systems and Computer Science (ICSCS), 2012 1st International Conference on*, pages 1–5, Aug 2012. doi: 10.1109/IConSCS.2012.6502449. URL http://dx.doi.org/10.1109/IConSCS.2012.6502449.

[Suleiman 2012] B. Suleiman. Elasticity Economics of Cloud-Based Applications. In *Proceedings of the 2012 IEEE Ninth International Conference on Services Computing*, SCC '12, pages 694–695, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4753-4. doi: 10.1109/SCC.2012.65. URL http://dx.doi.org/10.1109/SCC.2012.65.

[Tang et al. 2011] L. Tang, J. Mars, N. Vachharajani, R. Hundt, and M. L. Soffa. The impact of memory subsystem resource sharing on datacenter applications. *ACM SIGARCH Computer Architecture News*, 2011.

[Tinnefeld et al. 2014] C. Tinnefeld, D. Taschik, and H. Plattner. Quantifying the Elasticity of a Database Management System. In *DBKDA 2014, The Sixth International Conference on Advances in Databases, Knowledge, and Data Applications*, pages 125–131, 2014. URL http://www.thinkmind.org/index.php?view=article&articleid=dbkda_2014_5_30_50076.

[von Kistowski 2014] J. G. von Kistowski. Modeling Variations in Load Intensity Profiles. Master Thesis, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, 2014. URL http://sdqweb.ipd.kit.edu/publications/pdfs/Kistowski2014.pdf.

[von Kistowski et al. 2014] J. G. von Kistowski, N. R. Herbst, and S. Kounev. Modeling Variations in Load Intensity over Time. In *Proceedings of the 3rd International Workshop on Large-Scale Testing (LT 2014), co-located with the 5th ACM/SPEC International Conference on Performance Engineering (ICPE 2014)*. ACM, March 2014. URL http://dx.doi.org/10.1145/2577036.2577037.

[Weinman 2011] J. Weinman. Time is Money: The Value of "On-Demand", 2011. URL http://www.joeweinman.com/resources/Joe_Weinman_Time_Is_Money.pdf. (accessed July 9, 2014).

[Zhou et al. 2004] P. Zhou, V. Pandey, J. Sundaresan, A. Raghuraman, Y. Zhou, and S. Kumar. Dynamic tracking of page miss ratio curve for memory management. *ACM SIGOPS Operating Systems Review*, 2004.

[Zhuravlev et al. 2010] S. Zhuravlev, S. Blagodurov, and A. Fedorova. Addressing shared resource contention in multicore processors via scheduling. *Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems - ASPLOS '10*, 45, 2010.